Lecture 3  Announcements:

- HW 1 (due)
- SSH Key (due)
- Papercall (due)
- Talks of sign-up
- Pipeline mystery
SMT/“Hyperthreading”

Q: Potential issues?
Q: Potential issues?

- nx demand for memory across the hierarchy
- Predictability
- Power
Outline

Introduction
About This Class
Why Bother with Parallel Computers?
Lowest Accessible Abstraction: Assembly
Architecture of an Execution Pipeline

Architecture of a Memory System
Trends in Computer Architecture

Machine Abstractions

Performance Measurement
More bad news from Dennard

\begin{tabular}{|l|c|}
\hline
Parameter & Factor \\
\hline
Dimension & $1/\kappa$ \\
Line Resistance & $\kappa$ \\
Voltage drop & $\kappa$ \\
Response time & $1$ \\
Current density & $\kappa$ \\
\hline
\end{tabular}

[Dennard et al. ‘74, via Bohr ‘07]

- The above scaling law is for on-chip interconnects.
- Current $\sim$ Power vs. response time

Getting information from

- processor to memory
- one computer to the next

is

- slow (in \textit{latency})
- power-hungry
Somewhere Behind the Interconnect: Memory

Performance characteristics of memory:

- Bandwidth
- Latency

*Flops are cheap*

*Bandwidth is money*

*Latency is physics*

- M. Hoemann

Minor addition (but important for us)?

Bandwidth is money+ code structure
Latency is Physics: Distance

[Wikipedia ©️]
Latency is Physics: Electrical Model
Latency is Physics: DRAM
Latency is Physics: Performance Impact?

What is the performance impact of high memory latency?

Idea:

➤ Put a look-up table of recently-used data onto the chip.
➤ Cache
Memory Hierarchy

Registers

L1 Cache

L2 Cache

L3 Cache

DRAM

Virtual Memory (hard drive)

1 kB, 1 cycle

10 kB, 10 cycles

100 kB, 10 cycles

10 MB, 100 cycles

1 GB, 1000 cycles

1 TB, 1 M cycles
A Basic Cache

Demands on cache implementation:

- Fast, small, cheap, low power
- Fine-grained
- High “hit”-rate (few “misses”)

Design Goals: at odds with each other. Why?
Caches: Engineering Trade-Offs

Engineering Decisions:

- More data per unit of access matching logic
  → Larger “Cache Lines”
- Simpler/less access matching logic
  → Less than full “Associativity”
- Eviction strategy
- Size
Associativity

Direct Mapped:

<table>
<thead>
<tr>
<th>Memory</th>
<th>Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

2-way set associative:

<table>
<thead>
<tr>
<th>Memory</th>
<th>Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>
Miss rate versus cache size on the Integer portion of SPEC CPU2000 [Cantin, Hill 2003]
Demo: Learning about Caches

Demo: intro/Cache Organization on Your Machine
Experiments: 1. Strides: Setup

```c
int go(unsigned count, unsigned stride)
{
    const unsigned array_size = 64 * 1024 * 1024;
    int *ary = (int *) malloc(sizeof(int) * array_size);

    for (unsigned it = 0; it < count; ++it)
    {
        for (unsigned i = 0; i < array_size; i += stride)
            ary[i] *= 17;
    }

    int result = 0;
    for (unsigned i = 0; i < array_size; ++i)
        result += ary[i];

    free(ary);
    return result;
}
```

What do you expect? [Ostrovsky ‘10]
Experiments: 1. Strides: Results
Experiments: 2. Bandwidth: Setup

```c
int go(unsigned array_size unsigned steps)
{
    int *ary = (int *) malloc(sizeof(int) * array_size);
    unsigned asm1 = array_size - 1;

    for (unsigned i = 0; i < 100*steps;)
    {
        #define ONE ary[(i++)&16] ++;
        #define FIVE ONE ONE ONE ONE ONE
        #define TEN FIVE FIVE
        #define FIFTY TEN TEN TEN TEN TEN TEN
        #define HUNDRED FIFTY FIFTY FIFTY
        HUNDRED
    }

    int result = 0;
    for (unsigned i = 0; i < array_size; ++i)
        result += ary[i];

    free(ary);
    return result;
}
```

What do you expect? [Ostrovsky ‘10]
Experiments: 2. Bandwidth: Results
Experiments: 3. A Mystery: Setup

```c
int go(unsigned array_size, unsigned stride, unsigned steps)
{
    char *ary = (char *) malloc(sizeof(int) * array_size);

    unsigned p = 0;
    for (unsigned i = 0; i < steps; ++i)
    {
        ary[p] += p += stride;
        if (p >= array_size)
            p = 0;
    }

    int result = 0;
    for (unsigned i = 0; i < array_size; ++i)
    {
        result += ary[i];
    }
    free(ary);
    return result;
}
```

What do you expect? [Ostrovsky '10]
Experiments: 3. A Mystery: Results

Color represents achieved bandwidth:

- Red: high
- Blue: low
Thinking about the Memory Hierarchy

- What is a working set?
- What is data locality of an algorithm?
- What does this have to with caches?
Case Study: Streaming Workloads

Q: Estimate expected throughput for `saxpy` on an architecture with caches. What are the right units?

\[ z_i = \alpha x_i + y_i \quad (i = 1, \ldots, n) \]

Consider: mean bw 6 [bytes/sec]

\[ \text{bw} = \frac{3.4 \text{ bytes}}{\delta} \]
Case study: Matrix-Matrix Mult. (’MMM’): Code Structure

- How would you structure a high-performance MMM?
- What are sources of concurrency?
- What should you consider your working set?